Routing Multimedia Traffic with QoS Guarantees

Turgay Korkmaz[a] Marwan Krunz[b]

[a]Department of Computer Science
The University of Texas at San Antonio
6900 North Loop 1604 West, San Antonio, TX 78249
korkmaz@cs.utsa.edu
Phone: (210) 458-7346 Fax: (210) 458-4437

[b]Department of Electrical and Computer Engineering
University of Arizona
1230 E. Speedway, Tucson, AZ 85721
krunz@ece.arizona.edu
Phone(520) 621-8731 Fax:(520) 621-3862

Abstract— **One of the challenging issues in exchanging multimedia information over a network is how to determine a *feasible* path that satisfies all the quality-of-service (QoS) requirements of multimedia applications while maintaining high utilization of network resources. The latter objective implies the need to impose an additional optimality requirement on the feasibility problem. This can be done through a *primary* cost function (e.g., administrative weight, hop-count) according to which the selected feasible path is optimal. In general, multi-constrained path selection, with or without optimization, is an NP-complete problem that cannot be exactly solved in polynomial time. Heuristics and approximation algorithms with polynomial- and pseudo-polynomial-time complexities are often used to deal with this problem. However, existing solutions suffer either from excessive computational complexities that cannot be used for online network operation or from low performance. Moreover, they only deal with special cases of the problem (e.g., two constraints without optimization, one constraint with optimization, etc.). For the feasibility problem under multiple constraints, some researchers have recently proposed a *nonlinear* cost function whose minimization provides a continuous spectrum of solutions ranging from a generalized linear approximation (GLA) to an asymptotically exact solution. In this paper, we propose an efficient heuristic algorithm for the most general form of the problem. We first formalize the theoretical properties of the above *nonlinear* cost function. We then introduce our heuristic algorithm (H_MCOP), which attempts to minimize both the nonlinear cost function (for the feasibility part) and the primary cost function (for the optimality part). We prove that H_MCOP guarantees at least the performance of GLA and often improves upon it. H_MCOP has the same order of complexity as Dijkstra's algorithm. Using extensive simulations on random graphs and realistic network topologies with correlated and uncorrelated link weights from several distributions including uniform, normal, and exponential, we show the efficiency of H_MCOP over its (less general) contenders in terms of finding feasible paths and minimizing their costs under the same level of computational complexity.**

Keywords— **Multi-constrained path selection, QoS routing, scalable routing.**

EDICS Category: 5-BEEP

This work was supported by the National Science Foundation under grant ANI 9733143. An abridged version of this paper was presented at the *INFOCOM 2001 Conference*, Anchorage, Alaska, April 2001.

I. Introduction

Multimedia systems process and disseminate multiple time- and quality-dependent media such as video, voice, and data in three different forms [1]: standalone systems, systems connected via dedicated links, and systems connected via a wide-area network. While the popularity of the first two forms is increasing at a rapid rate, the last form cannot be widely used due to the shortcomings of the current best-effort Internet technology. However, the continuous demand for exchanging multimedia information over the Internet is calling for new networking services (e.g., Diffserv, Intserv, MPLS, ATM) that are geared towards providing quality-of-service (QoS) guarantees. To support the expected functionalities of distributed multimedia applications, multimedia information needs to be transfered over the network with the guarantee of several QoS requirements (e.g., bandwidth, delay, jitter, reliability). As shown in Figure 1, two multimedia systems can communicate using one of the several alternative paths in the network. Since each path can only guarantee a certain

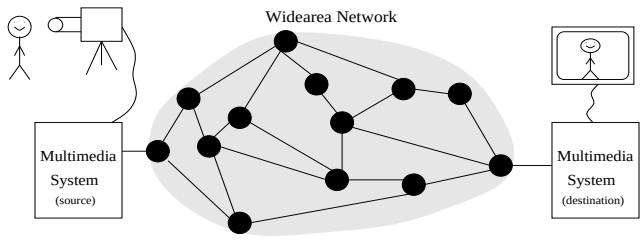

Fig. 1. Example of a distributed multimedia application.

level of QoS, network operators need to identify a feasible path that satisfies the QoS requirements of the application while simultaneously achieving efficient utilization of network resources. This problem of finding a feasible, pos-

sibly optimal, path subject to several constraints is known as QoS (or constraint-based) routing [2], [3], [4], [5], [6], [7]. QoS routing is the first step that needs to be done in both reservation-based services (e.g., Intserv, MPLS, ATM) as well as reservationless services (e.g., Diffserv). For example, in the ATM PNNI protocol [8], constraint-based routing is performed by source nodes to determine suitable paths for connection requests. Similarly, in MPLS, which is a convergence of several efforts aimed at combining the best features of IP and ATM [9], a source router selects a path subject to QoS requirements and uses a signaling protocol (e.g., RSVP or CR-LDP) to reserve resources along that path. For QoS routing, routers need information about available network resources [9]. The mechanism for acquiring such information can be part of the path establishment protocol, as in the case of PNNI, or it may be provided via existing routing protocols, as in the case of MPLS. For example, MPLS relies on protocols such as OSPF (Open Shortest Path First) [10] to provide the link state information (e.g., delay, bandwidth, etc.). This requires slight extension of OSPF, as described in [11]. The type-of-service (TOS) field in OSPF, which has not been much used in the past, has now been redefined to advertise multiple link parameters (see [11] for details). In the case of Diffserv, the constraint-based routes can be requested, for example, by network administrators for traffic engineering purposes. Provisioning of such routes can also be used to guarantee a certain service level agreement (SLA) for aggregated flows [7].

In general, routing consists of two basic tasks: distributing the state information of the network and searching this information for a feasible, possibly optimal path. In this paper, we focus on the second task and assume that the true state of the network is available to every node (e.g., via link-state routing protocols such as OSPF) and that nodes use this information to determine end-to-end paths between distributed multimedia systems (see [12] for QoS routing under inaccurate information). Each link in the network is associated with multiple QoS parameters which can be roughly classified into additive and non-additive [13], [14]. For the additive parameters (e.g., delay, jitter, administrative weight), the cost of an end-to-end path is given by the *sum* of the individual link values along that path[1]. In contrast, the cost of a path with respect to (w.r.t.) a non-additive parameter, such as bandwidth, is determined by the value of that constraint at the bottleneck link. It is known that constraints associated with non-additive parameters can be easily dealt with a preprocessing step by pruning all links that do not satisfy these constraints [15]. Hence, in this paper we will mainly focus on additive QoS parameters and assume that the optimality of a path is evaluated based on an additive parameter (e.g., administrative weight, hop-count). The underlying problem can be stated as follows.

Definition 1 *Multi-Constrained Optimal Path (MCOP) Problem*: Consider a network that is represented by a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of links. Each link $(i, j) \in E$ is associated with a primary cost parameter $c(i, j)$ and K additive QoS parameters $w_k(i, j)$, $k = 1, 2, \ldots, K$; all parameters are non-negative. Given K constraints c_k, $k = 1, 2, \ldots, K$, the problem is to find a path p from a source node s to a destination node t such that:
(i) $w_k(p) \stackrel{\text{def}}{=} \sum_{(i,j)\in p} w_k(i, j) \leq c_k$ for $k = 1, 2, \ldots, K$, and
(ii) $c(p) \stackrel{\text{def}}{=} \sum_{(i,j)\in p} c(i, j)$ is minimized over all feasible paths satisfying (i).

For $K = 1$ the MCOP problem is known as the restricted shortest path (RSP) problem, which is NP-complete [16]. A slightly different version of the MCOP problem is known as the multi-constrained path (MCP) problem, which aims at finding *any* feasible path w.r.t. multiple constraints (no path optimization is done). For $K \geq 2$ the MCP problem is also known to be NP-complete [17], [18]. Both the RSP and MCP problems can be solved via pseudo-polynomial-time algorithms whose complexities depend on the actual values of the link weights (e.g., maximum link weight) in addition to the size of the network [19], [18]. However, these algorithms are computationally expensive if the values of the link weights are large. To cope with the NP-completeness of these problems, researchers have resorted to several heuristics and approximation algorithms.

For the RSP problem, several different solutions are proposed in the literature. The author in [20] proposed the Constrained Bellman-Ford (CBF) algorithm. Although this algorithm exactly solves the RSP problem, its running time grows exponentially in the worst case. The author in [19] presented two ϵ-optimal approximation algorithms with the complexities of $\mathcal{O}(\log\log B(m(n/\epsilon)) + \log\log B)$ and $\mathcal{O}(m(n^2/\epsilon)\log(n/\epsilon))$, where B is an upper bound on the solution (e.g., the longest path), n is the number of nodes, m is the number of links, and ϵ shows how far the solution from the optimal. The basic idea behind these approximations is to bound input data through rounding and scaling it and then to use a pseudo-polynomial-time algorithm. In [21] the author provided another ϵ-optimal approximation with the complexity of $\mathcal{O}(mn(1+1/\epsilon)+n^2(1+1/\epsilon)(\log n+\log(1+1/\epsilon)))$. In [22] the authors considered a related problem, in which it is needed to find the least-cost path from a given source to all destinations while satisfying the given constraint. For this problem, the authors provided an ϵ-approximation algorithm with the complexity of $\mathcal{O}((m + n\log n)D/\epsilon)$, where D can be at most $n - 1$. In [23] the authors proposed the ϵ-optimal approximation algorithm for a RSP-related problem, in which one link weight is a function of the other. The above approximation algorithms provide better performance in minimizing the cost of returned feasible paths as ϵ goes to zero. However, small values of ϵ make the computational complexity of such algorithms prohibitive for online network operation. Accordingly, the author in [24], [25] modified ϵ-optimal approximation algorithms in [19] to provide better scalability in hierarchical networks. Another approach to the RSP problem is to find the k-shortest paths w.r.t. a cost function

[1]Multiplicative constraints, such as link reliability, can be transformed into additive constraints.

defined based on the combination of link weights and the given constraint, hoping that one of these paths is feasible and near-optimal [26], [27], [28]. The value of k determines the performance and overhead of this approach; if k is large, the algorithm has good performance but its computational cost is expensive. A similar approach to the k-shortest paths is to *implicitly* enumerate all feasible paths [29], but this approach is also computationally expensive. The authors in [30] proposed a distributed heuristic solution for RSP, called the delay-constrained unicast routing (DCUR) algorithm. Its message complexity is $\mathcal{O}(n^2)$, where n is the number of nodes. The basic idea is to explore the graph based on the concatenation of two segments: (1) the so-far explored path from the source s to a node u; (2) the least-delay or the least-cost path from the node u to the destination t. In [31] the authors considered the same DCUR algorithm and provided some improvement over the original DCUR by reducing the message complexity to $\mathcal{O}(n)$. In [32] the authors considered a similar algorithm to DCUR and discussed its use in multipath routing [33]. In [34] the authors provided a distributed algorithm based on probing and backtracking. In addition to the delay and cost, this algorithm considers bandwidth as well. Several researchers investigated Lagrangian-based search in which the single link weight (e.g., delay) and the cost are linearly combined as a single metric, hoping that the shortest path w.r.t. this single metric is feasible while minimizing the cost. A key issue here is how to determine the appropriate weights (or multipliers) while combining the delay and cost. In [35] the authors proposed a systematic way of searching for appropriate weights to combine the delay and the cost as follows. The algorithm finds the shortest path according to the current linear combination of the delay and the cost. Using this shortest path, it then adjusts the weights of the delay and the cost in the linear combination and repeats itself to approach the optimal path. The authors showed that this search takes finite iterations of Dijkstra's algorithm assuming that paths are uniformly distributed in the delay-cost space. The same Lagrangian-based search with some extensions was also considered by others (e.g., [28], [36], [37]). For example, the authors in [28] considered the k-shortest path algorithm to close the gap between the optimal solution and the returned path based on the linear combination. Although the computational results indicate order of magnitude savings, the amount of time to determine an optimal path may be excessive in some cases. In [37], the authors showed that the worst-case complexity of the above algorithm is $\mathcal{O}(m^2 \log^2(m))$, i.e., polynomial time. The above algorithms are especially proposed for the RSP problem (i.e., they do not consider multiple constraints) and their computational complexities are often excessive in the worst case.

In [18] the author considered the MCP problem under two constraints and proposed an intuitive approximation algorithm based on minimizing a linear combination of the link weights. More specifically, this algorithm returns the shortest path w.r.t. $l(e) \stackrel{\text{def}}{=} \alpha w_1(p) + \beta w_2(p)$ by using Dijkstra's shortest path algorithm, where $\alpha,\ \beta \in Z^+$. The key issue here is to determine the appropriate α and β such that an optimal path w.r.t. $l(e)$ is likely to satisfy the individual constraints. Based on minimizing an objective function of the form $f(p) = \max\{w_1(p), c_1\} + \max\{w_2(p), c_2\}$, the author in [18] provided two sets of values for α and β: (a) $\alpha = \beta = 1$; (b) $\alpha = 1$ and $\beta = \sqrt{c_1/c_2}$. Using either one of these sets as a good guess, the algorithm computes the shortest path w.r.t. the above linear combination. If the returned path is feasible w.r.t. both constraints, then the algorithm succeeds in finding a feasible path within the complexity of Dijkstra's algorithm. However, if the returned path is not feasible, then the algorithm cannto proceed. Actually, the search can continue by using different values for α and β, and can lead to finding a feasible path. In [38], we invastigated this approach and proposed an approximation algorithm called Approx_2CP that dynamically searches for approapriate values of α and β with the complexity of $\mathcal{O}(\log B(m + n \log n))$. Another heuristic for the MCP problem under two constraints was proposed in [39]. In this study, the original problem was modified by scaling down the values of one of the two link weights to bounded integers. It was shown that the modified problem can be solved by using Dijkstra's (or Bellman-Ford) shortest path algorithm and that the solution to the modified problem is also a solution to the original one. When Dijkstra's algorithm is used, the computational complexity of the algorithm is $\mathcal{O}(x^2 n^2)$; when Bellman-Ford algorithm is used, the complexity is $\mathcal{O}(xnm)$, where x is an adjustable positive integer whose value determines the performance and overhead of the algorithm. To achieve a high probability of finding a feasible path, x needs to be as large as $10n$, resulting in computational complexity of $\mathcal{O}(n^4)$. In [40] this heuristic algorithm is generalized to more than two constraints with the complexity of $\mathcal{O}(x_1^2 \cdot \ldots \cdot x_{K-1}^2 n^2)$ or $\mathcal{O}(x_1 \cdot \ldots \cdot x_{K-1} nm)$, where $x_1, \ldots, x_{K-1}$ are adjustable integers for each constraint. In [41] the authors extended Bellman-Ford algorithm to address the problem under two constraints. For finding feasible paths under *an arbitrary* number of constraints, we proposed a randomized algorithm (called R_MCP) in [42]. The complexity of R_MCP is $K + 2$ times that of Dijkstra's algorithm, where K is the number of additive constraints. The authors in [43] also addressed the MCP problem under an arbitrary number of constraints, and provided an algorithm (called TAMCRA) using the k-shortest path algorithm in [44] along with a nonlinear cost function. The complexity of TAMCRA is $\mathcal{O}(km \log(kn) + k^3 m)$, where k is the number of shortest paths. As mentioned above, the performance and overhead of this algorithm depend on k. However, our simulation results show that even with small values of k, the algorithm gives good performance in finding feasible paths. However, the above algorithms are especially proposed for the MCP problem (i.e., they do not attempt to optimize the selection of the feasible path).

Other works in the literature were aimed at addressing special yet important cases of the QoS routing problem. For example, several researchers addressed the QoS routing in the context of bandwidth and delay param-

eters. Showing that the feasibility problem under this combination is not NP-complete, the authors in [45] presented a *bandwidth-delay based routing algorithm* which simply prunes all links that do not satisfy the bandwidth requirement and then finds the shortest path w.r.t. delay in the pruned graph. Several path selection algorithms based on different combinations of bandwidth, delay, and hop-count were discussed in [46], [47] (e.g., widest-shortest path, shortest-widest path). In addition, new algorithms were proposed to find more than one feasible path w.r.t. bandwidth and delay (e.g., Maximally Disjoint Shortest and Widest Paths) [48]. In [49] the authors proposed bandwidth guaranteed dynamic routing algorithms. In [50] the authors considered pre-computation of paths with minimum hop-count and bandwidth guarantee. They also provided some approximation algorithms that takes into account general additive constraints during the pre-computation. In [51] the authors investigated how to set link weights based on the previous measurements so that the shortest paths can provide better load balancing and can meet the desired QoS requirements. Some researchers considered the fallback routing approach [52], [53], in which QoS parameters are ordered and the optimal path w.r.t. each single parameter in this order is found until the returned path is feasible w.r.t. all constraints. Another approach to QoS routing is to exploit the dependencies between the QoS parameters and solve the path selection problem assuming specific scheduling schemes at network routers [54], [55]. Specifically, if Weighted Fair Queueing (WFQ) scheduling [56], [57], [58] is being used and the constraints are bandwidth, queueing delay, jitter, and loss, then the problem can be reduced to standard shortest path problem by representing all the constraints in terms of bandwidth. Although queueing delay can be formulated as a function of bandwidth, this is not the case for the propagation delay, which needs to be taken into account for QoS routing in high-speed networks [59].

Contributions and Organization of the Paper

As reviewed above, previously proposed algorithms consider only special cases of the MCOP problem (e.g., RSP, MCP) and suffer from either excessive computational complexities or low performance. In this paper, we provide an efficient heuristic algorithm to the most general form of the MCOP problem. By general, we mean that our solution is applicable to any number of constraints, irrespective of their nature and interdependence, and the optimization of the selected feasible path. In Section II, to deal with multiple constraints, we consider the same nonlinear cost function provided in [43]. We show that the minimization of this cost function, which involves a predetermined constant λ, gives new insights into finding a feasible path in the MCP problem by offering a continuous spectrum of solutions ranging from a simple, linear approximation ($\lambda = 1$) to an asymptotically optimal solution ($\lambda \to \infty$). Note that the authors in [43] have used the k-shortest path algorithm to minimize the nonlinear function for the MCP problem (without path optimization). In Section III, we introduce a new efficient heuristic algorithm (H_MCOP) to minimize the nonlinear cost function for finding a feasible path while also incorporating the optimization of the selected feasible path. We prove that H_MCOP guarantees at least the performance that can be obtained by a generalized linear approximation algorithm and often improves upon it. The computational complexity of H_MCOP is two times that of Dijkstra's algorithm. We note that, like TAMCRA, H_MCOP can also be used with k-shortest path algorithm to improve performance further. Using extensive simulations, we compared several algorithms in Section IV. Simulation results indicate that H_MCOP and TAMCRA gives better performance than other algorithms in finding feasible paths under the same order of complexity. Moreover, H_MCOP significantly improves the optimality of the obtained feasible paths over TAMCRA.

II. Nonlinear Cost Function for MCP

Consider the following cost function for any path p from the source to the destination:

$$g_\lambda(p) \stackrel{\text{def}}{=} (\frac{w_1(p)}{c_1})^\lambda + (\frac{w_2(p)}{c_2})^\lambda + \cdots + (\frac{w_K(p)}{c_K})^\lambda \qquad (1)$$

where $\lambda \geq 1$. Suppose there is an algorithm $\mathcal{X}$ that returns a path p by minimizing the cost function (1) for a given $\lambda \geq 1$. Then, the following bounds on the performance of algorithm $\mathcal{X}$ can be established.

Theorem 1: Consider the MCP problem (i.e., the MCOP problem without optimizing the selection of a feasible path). Assume that there is at least one feasible path $p*$ in the network. Let p be a path that minimizes the cost function g_λ for a given $\lambda \geq 1$. Then, (i) $w_k(p) \leq c_k$ for at least one k, and (ii) $w_k(p) \leq \sqrt[\lambda]{K} c_k$ for all other k's.

Proof: If the returned path p is feasible, then from (1) the above bounds are correct. Assume that p is not feasible. Since the algorithm returns the path p (and not the feasible path $p*$), it must be true that

$$g_\lambda(p) \leq g_\lambda(p*)$$

In addition, since $w_k(p*) \leq c_k$ for all k's, we have

$$g_\lambda(p*) \leq K$$

Thus,

$$g_\lambda(p) \leq K \qquad (2)$$

If $w_k(p) > c_k$ for all k's, then $g_\lambda(p) > K$. Since this contradicts (2), we must have $w_k(p) \leq c_k$ for at least one k, and the bound in part (i) is correct. Note that if $g_\lambda(p) > K$, then it is guaranteed that there is no feasible path $p*$ in G because for at least one k, $w_k(q) > c_k$ for every path q.

To prove part (ii), assume to the contrary that for at least one constraint c_j we have $w_j(p) > \sqrt[\lambda]{K} c_j$, so that $(\frac{w_j(p)}{c_j})^\lambda > K$. It readily follows that $g_\lambda(p) \geq (\frac{w_j(p)}{c_j})^\lambda > K$, which contradicts (2). Hence, part (ii) is proved. ■

Corollary 1: As λ increases, the likelihood of finding a feasible path also increases.

Proof: Follows immediately from Theorem 1 ($w_k(p) \leq \sqrt[\lambda+\delta]{K}c_k < \sqrt[\lambda]{K}c_k$, for any $\delta > 0$). ■

Therefore, to increase the probability of finding a feasible path, it makes sense to set λ to its largest possible value, i.e., $\lambda \to \infty$. In order to provide a practical computational model for $\lambda \to \infty$, we can replace the cost function $g^*(p) \stackrel{\text{def}}{=} \lim_{\lambda\to\infty} g_\lambda(p)$ by another cost function that does not explicitly involve λ but that achieves the same ordering of candidate paths as g^*. More precisely, we consider the following cost function for a path p [43]:

$$h(p) \stackrel{\text{def}}{=} \max\{\frac{w_1(p)}{c_1}, \frac{w_2(p)}{c_2}, \cdots, \frac{w_K(p)}{c_K}\} \quad (3)$$

The following theorem establishes the equivalence of g^* and h.

Theorem 2: Let p_1 and p_2 be any two paths. Then $g^*(p_1) \leq g^*(p_2)$ iff $h(p_1) \leq h(p_2)$.

Proof: It is obvious that as $\lambda \to \infty$, $g_\lambda(p)$ is dominated by the largest term in (1), or equivalently, by $\max\{\frac{w_1(p)}{c_1}, \frac{w_2(p)}{c_2}, \cdots, \frac{w_K(p)}{c_K}\}$. A similar argument can be used to establish the proof in the other direction. ■

Figure 2 depicts a pictorial illustration of how algorithm $\mathcal{X}$ finds a feasible path with three different values of λ. The shaded area represents the feasibility region in the 2D parameter space (i.e., two weights are associated with each link). The black dots represent the normalized costs of various paths w.r.t. w_1 and w_2. Each contour line in the figure indicates paths with equal value w.r.t. the given cost function. Starting at the origin, algorithm $\mathcal{X}$ slides the fixed-cost contour line outward in the direction of the arrow until it hits a path (i.e., black dot in the figure). The returned path has the minimum cost w.r.t. g_λ. As shown in the figure, the larger the λ value, the closer the shape of the contour lines to that of the (hypercube) feasibility region. As $\lambda \to \infty$, algorithm $\mathcal{X}$ becomes exact, i.e., it is guaranteed to find a feasible path if one exists.

For $\lambda = 1$ it is easy to develop a polynomial-time algorithm that minimizes $g_1(p)$. This is done by assigning a combined weight $l(e) = \frac{w_1(e)}{c_1} + \frac{w_2(e)}{c_2} + \cdots + \frac{w_K(e)}{c_K}$ to every link e and finding the shortest path w.r.t. $l(e)$ using Dijkstra's algorithm. From Theorem 1, when $\lambda = 1$, the returned path p satisfies the following bounds: (i) $w_k(p) \leq c_k$ for at least one k, and (ii) $w_k(p) \leq Kc_k$ for all other k's. As a matter of fact, this is a generalized linear approximation algorithm that applies to any number of constraints. It includes as special cases the approximation algorithms developed in [18] (except for the normalization factors).

For $\lambda > 1$ the nonlinearity of (1) makes it impossible to provide an exact polynomial-time minimization algorithm. Hence, one has to rely on heuristics. One such heuristics (the TAMCRA) was proposed in [43], which aims only at finding a feasible path based on the k-shortest path algorithm. Since no path optimization is performed, the selected path, albeit feasible, may be undesirable from a cost standpoint. Our goal is to provide a new heuristic algorithm that addresses both the feasibility aspect as well as the cost effectiveness of the selected path.

Due to the heuristic nature of our algorithm, its performance may not always improve monotonically with λ, i.e., it is possible that the heuristic search fails to find a feasible path based on λ_2, which otherwise can be found based on $\lambda_1 < \lambda_2$). However, simulation results show that increasing λ most often results in performance improvement. Hence, it makes sense to base the design of our heuristic on the cost function g^*, or equivalently h.

III. Proposed Heuristic for MCOP

We now present our heuristic algorithm H_MCOP, which attempts to find a feasible path subject to K additive constraints and, simultaneously, minimize the cost of that path. For the feasibility part, H_MCOP tries to minimize the objective function g_λ for $\lambda > 1$. In doing so, it first exactly finds the best path w.r.t. g_1 from each node u to t. It then starts from s and discovers each node u based on the minimization of $g_\lambda(P)$, where P is a complete s-t path passing through node u. This s-t path is heuristically determined at node u by concatenating the already traveled segment from s to u and the estimated remaining segment (the above best path w.r.t. g_1) from u to t. Since the algorithm considers complete paths, it can foresee several paths before reaching the destination. For the optimality part, If some of these foreseen paths are feasible, H_MCOP selects the one that minimizes the primary cost function rather than the one that minimizes the nonlinear cost function. Using this preference rule (i.e., minimize the primary cost function if the foreseen path is feasible; otherwise, minimize the nonlinear cost function), H_MCOP can be implemented as simple as single-objective algorithms.

A pseudocode for H_MCOP is shown in Figure 3. Its inputs are a directed graph $G = (V, E)$ in which each link (i, j) is associated with a primary cost $c(i, j)$ and K weights $w_k(i, j)$, $k = 1, 2, \ldots, K$; a source node s; a destination node t; and K constraints c_k, $k = 1, 2, \ldots, K$. For each node u, the algorithm maintains the following labels: $r[u]$, $R_k[u]$, $\pi_r[u]$, $g[u]$, $G_k[u]$, $\pi_g[u]$, and $c[u]$, $k = 1, 2, \ldots, K$. Label $r[u]$ represents the cost of the shortest path from u to t w.r.t. the cost function g_1 (i.e., g_λ with $\lambda = 1$). Labels $R_k[u]$, $k = 1, 2, \ldots, K$, represent the individually accumulated link weights along that path. The predecessor

```
H_MCOP(G = (V, E), s, t, c_k, k = 1, 2, ..., K)
1  Reverse_Dijkstra(G = (V, E), t);
2  if r[s] > K then
3      return failure // there is no feasible path
4  end if
5  Look_Ahead_Dijkstra(G = (V, E), s);
6  if G_k[t] ≤ c_k ∀ k = 1, 2, ..., K then
7      return the path // a feasible path is found
8  end if
9  return failure   // a feasible path does not exist
                    // or H_MCOP cannot find it
```

Fig. 3. The heuristic algorithm H_MCOP for the MCOP problem.

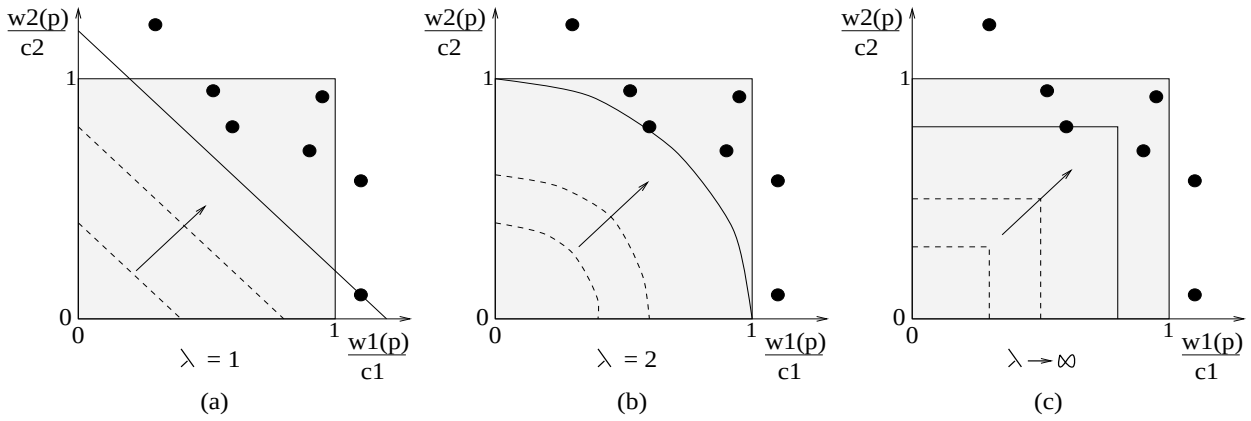

Fig. 2. Searching for a feasible path that minimizes $g_\lambda(p)$.

of u on this optimal path is stored in label $\pi_r[u]$. Label $g[u]$ represents the cost of a foreseen complete path that goes from s to t via node u w.r.t. the cost function h (or g_λ, $\lambda > 1$). Labels $G_k[u]$, $k = 1, 2, \ldots, K$, and $c[u]$ represent the individually accumulated cost of link weights and the primary cost along the already traveled segment of this path from s to u. The predecessor of u on this path is stored in the label $\pi_g[u]$.

There are two directions in the algorithm: backward (from t to s) to estimate the cost of the remaining segment using $\lambda = 1$ and forward (from s to t) to find the most promising path in terms of feasibility and optimality using $\lambda > 1$. In the backward direction (lines 1-4 in H_MCOP), the algorithm finds the optimal path from every node u to t w.r.t. the cost function $g_1(\cdot)$. For that, it uses Reverse_Dijkstra [16] with some modifications to the relaxation procedure, as shown in Figure 4. Note that

Reverse_Dijkstra_Relax(u, v)
1 **if** $r[u] > \sum_{k=1}^{K} \frac{R_k[v]+w_k(u,v)}{c_k}$ **then**
2 $\quad r[u] := \sum_{k=1}^{K} \frac{R_k[v]+w_k(u,v)}{c_k}$
3 $\quad R_k[u] := R_k[v] + w_k(u, v)$ for $k = 1, 2, \ldots, K$
4 $\quad \pi_r[v] := u$
5 **end if**

Fig. 4. Modified relaxation procedure for Reverse_Dijkstra based on minimizing $g_1(\cdot)$.

H_MCOP does not use $\pi_r[\cdot]$, the predecessors of nodes in the reverse paths from every node to t. They are just included for the completeness. Reverse_Dijkstra initially sets $r[u] = \infty$ and $\pi_r[u] = NIL$ for every node u. It then starts at node t by setting $r[t]$ and $R_k[t]$, $k = 1, 2, \ldots, K$, to zeros. It explores the graph and eventually returns a path p from s to t. Before proceeding further, the algorithm checks $r[s] > K$ to determine the possibility of discovering a feasible path (based on the proof of Theorem 1, $r[s] > K$ implies necessarily the nonexistence of a feasible path).

If there is a possibility that the network contains a feasible path, a heuristic search procedure called Look_Ahead_Dijkstra is executed in the forward direction (line 5 in H_MCOP). This procedure uses the information provided by the above Reverse_Dijkstra to identify whether there is another path q which *provably* improves the performance over the above returned path p. To implement Look_Ahead_Dijkstra, we need to slightly modify the relaxation process of Dijkstra's algorithm [60], as shown in Figure 5.

Look_Ahead_Dijkstra_Relax(u, v)
1 Let tmp be a temporary node
2 $c[tmp] := c[u] + c(u, v)$
3a **if** $\lambda < \infty$ **then**
4a $\quad g[tmp] := \sum_{k=1}^{K} (\frac{G_k[u]+w_k(u,v)+R_k[v]}{c_k})^\lambda$
5a **end if**
3b **if** $\lambda = \infty$ **then**
4b $\quad g[tmp] := \max\{\frac{G_k[u]+w_k(u,v)+R_k[v]}{c_k} \mid 1 \le k \le K\}$
5b **end if**
6 $G_k[tmp] := G_k[u] + w_k(u, v)$ for $k = 1, 2, \ldots, K$
7 $R_k[tmp] := R_k[v]$ for $k = 1, 2, \ldots, K$
8 **if** Prefer_the_best$(tmp, v) = tmp$) **then**
9 $\quad c[v] := c[tmp]$
10 $\quad g[v] := g[tmp]$
11 $\quad G_k[v] := G_k[tmp]$ for $k = 1, 2, \ldots, K$
12 $\quad \pi_g[v] := u$
13 **end if**

Fig. 5. Modified relaxation procedure for Look_Ahead_Dijkstra of H_MCOP.

Look_Ahead_Dijkstra initially sets $g[u] = \infty$ and $\pi_g[u] = NIL$ for every node u. It then starts from node s, setting $g[s]$, $c[s]$, and $G_k[s]$, $k = 1, 2, \ldots, K$, to zeros. It explores the graph by choosing the next node based on the preference rule in Figure 6. This rule takes as input two nodes and their labels. It then selects one of these nodes such that the selected one minimizes the primary cost function if foreseen s-t paths passing through these nodes are feasible; otherwise, it selects the one that minimizes the objective function g_λ.

Eventually, H_MCOP returns a path q from s to t using $\lambda > 1$. The following theorem guarantees that q cannot be

```
Prefer_the_best(a, b)
1  if c[a] < c[b] and ∀k  G_k[a] + R_k[a] ≤ c_k then
2      return(a)
3  end if
4  if c[a] > c[b] and ∀k  G_k[b] + R_k[b] ≤ c_k then
5      return(b)
6  end if
7  if g[a] < g[b] then return(a)
8  return(b)
```

Fig. 6. Preference rule used in H_MCOP to choose between two nodes a and b.

worse than the path p found using $\lambda = 1$, i.e., q has either less primary cost than p if p is feasible, or it has more chance of being feasible than p if p is not feasible. This theorem also states that H_MCOP guarantees at least the performance of the linear approximation algorithm with $\lambda = 1$ and often improves upon it.

Theorem 3: Suppose that H_MCOP algorithm returns the path p by searching backward from t to s (using $\lambda_1 = 1$) and, subsequently, returns the path q by searching forward from s to t (using $g_{\lambda_2}()$ with $\lambda_2 > 1$). Then, (i) if p is feasible, q is feasible and $c(q) \leq c(p)$; (ii) if p is not feasible, $g_{\lambda_2}(q) \leq g_{\lambda_2}(p)$.

Proof: Assume that p consists of l nodes $(v_0, v_1, v_2, \ldots, v_l)$ where $v_0 = s$ and $v_l = t$. In the forward direction, H_MCOP initially extracts s from heap and discovers its neighbors including v_1 and inserts them into the heap. H_MCOP then extracts the next node (say u) from the heap based on the preference rule given by the procedure Prefer_the_best. Actually this works as follows. Assume that we have $h_f + h_i$ nodes in the heap, where h_f is the number of nodes at which the foreseen paths are feasible (i.e., $\forall_k\, G_k[\cdot] + R_k[\cdot] \leq c_k$) and h_i is the number of nodes at which the foreseen paths are infeasible. If $h_f > 0$, then the next node is the one with minimum $c[\cdot]$ among these h_f nodes. Otherwise (i.e., $h_f = 0$), the next node is the one with minimum $g[\cdot]$.

Now first assume that p is feasible. In other words, $h_f > 0$ since at least the foreseen path at v_1 is feasible. If there is no other foreseen feasible path, then H_MCOP extracts v_1 and continue to explore the graph by visiting nodes $v_2, v_3, \ldots, v_l$. If there are some other nodes at which the foreseen paths are feasible, then H_MCOP extracts one of them with minimum $c[\cdot]$ and explores the graph through this node. As a result, H_MCOP returns either p or another feasible path q with less cost. Thus, part (i) is correct.

If all foreseen paths are infeasible, then the algorithm explores the graph based on the minimum $g[.]$. Again the algorithm considers v_1 at the first time. If $g[v_1]$ is minimum, then the algorithm will explore the graph from v_1 and continue to visit $v_2, v_3, \ldots, v_l$ as long as they have the minimum $g[\cdot]$. Otherwise, the algorithm will explore another node whose $g[\cdot]$ value is less than $g[v_1]$. As a result, H_MCOP returns either p or another path q with less $g(\cdot)$. Thus, part (ii) is also correct. Therefore, the returned path q will be either better than p or at least as good as p in terms of both feasibility and optimality. ■

The computational and space complexities of the resulting H_MCOP algorithm are equal to that of Dijkstra's, since *at most* two modified versions of Dijkstra's algorithm. To improve the performance, the forward direction of H_MCOP can be used with the k-shortest path implementation of Dijkstra's algorithm presented in [44]. In addition, as in TAMCRA, dominated paths can be eliminated to improve the performance. In this case, H_MCOP has the same worst-case complexity of TAMCRA in the forward direction. Our simulation results (shown in the next section) indicate that for the small values of k H_MCOP gives slightly better performance than TAMCRA in terms of finding feasible paths. Furthermore, due to the additional path optimization feature of H_MCOP, its returned paths are much more resource efficient than their TAMCRA counterparts.

Example

The following example illustrates the operation of H_MCOP under the nonlinear cost function h. For simplicity, we exclude path optimization from this example. Consider the network in Figure 7(a). For simplicity, assume that each link has two weights w_1 and w_2 and that links are symmetric (note that H_MCOP can run on asymmetric links with multiple real-valued weights). Suppose that a path is to be found from s to t which satisfies the constraints $c_1 = 10$ and $c_2 = 10$. Figure 7 describes the steps taken by H_MCOP to discover such a path. In the backward direction (Figures 7(a)-(c)), Reverse_Dijkstra finds a path from every node to the destination node t. Since the returned path (s, u, v, t) is not feasible and the value of the cost function ($r[s] = 1.6$) is less than $K = 2$, the algorithm proceeds to search for a feasible path in the forward direction using Look_Ahead_Dijkstra. Although Reverse_Dijkstra cannot find a feasible path in this example, it provides useful information (labels $R_k[u]$) for Look_Ahead_Dijkstra, enabling it to find a feasible path. Figures 7(d)-(e) show the state of the algorithm during the execution of Look_Ahead_Dijkstra based on the cost function h. The algorithm starts from s and discovers its neighbors u and v by relaxing links (s, u) and (s, v). The process of relaxing a link (i, j) consists of testing whether the cost of the foreseen path that goes through j can be improved by going through i to j and, if so, of updating the new values of the cost function and the predecessor of node j [60]. The algorithm then selects node v at which the value of the cost function is minimum and tries to discover its neighbors. Since the value of the cost function at node t decreases if link (v, t) is used, the algorithm relaxes this link. However, it cannot relax link (v, u) since the value of the cost function at node u does not decrease if this link is used. Now there are two nodes u and t to explore the graph. Since the value of the cost function at t is minimum, the algorithm selects it but cannot relax any more links. Finally, the algorithm selects u and relaxes

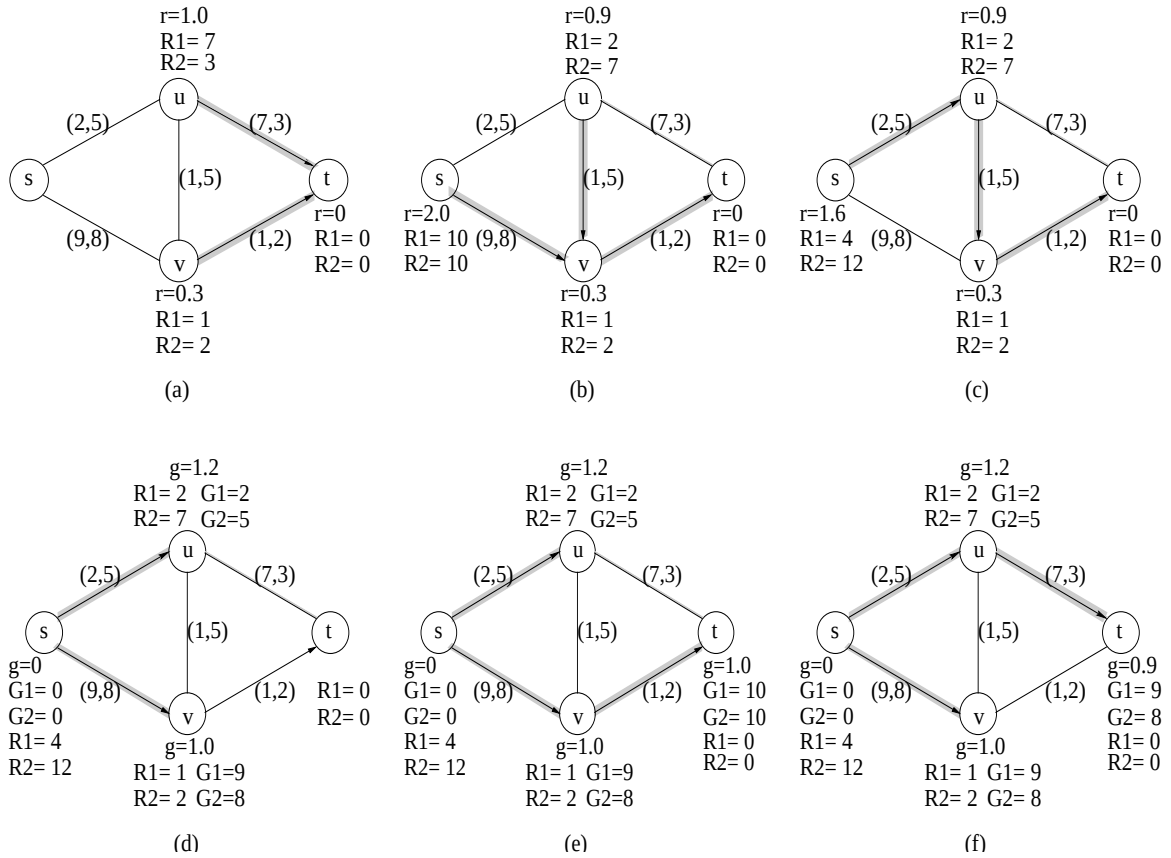

Fig. 7. Example that illustrates the operation of H_MCP. Shaded edges indicate the selected path. In (a)-(c) Reverse_Dijkstra returns an infeasible path. In (d)-(e) a feasible path is discovered using Look_Ahead_Dijkstra.

only the link (u, t) since the value of the cost function at t is decreased through this relaxation. Finally, the algorithm returns the feasible path (s, u, t).

IV. Performance Evaluation

In this section, we contrast the performance of H_MCOP with Jaffe's two approximation algorithms [18] (one with $\alpha = \beta = 1$ while the other with $\alpha = 1$ and $\beta = \sqrt{c_1/c_2}$)), Chen's heuristic algorithm [39], TAMCRA [43] and our other algorithms Approx_2CP in [38], R_MCP and its deterministic version ER_MCP_D in [42]. Note that our three algorithms have been designed with an increasing level of generality while improving the performance and the computational complexity. As a result of this effort, H_MCOP gives better performance than the others under the same order of computational complexity. In addition, H_MCOP tries to optimize the cost of the selected feasible paths. So we mainly focus on H_MCOP and compare it with various algorithms. In our simulations, several random network topologies have been studied and similar trends have been observed across these topologies.

A. Simulation Model and Performance Measures

In our simulations, we use the random graph generator package described in [61] to generate 50-, 100-, and 200-node random topologies based on Waxman's model [62]. In addition, we use two realistic topologies, shown in Figure 14. We associate two randomly generated weights (e.g., delay and delay-jitter) with each link (i, j). As shown in Table I, the two weights are sampled from two different uniform distributions with three levels of correlations (positively correlated, no correlation, and negatively correlated link weights). Positive correlation means that both weights have small means or large means. Negative correlation means that one of weights is selected from a uniform distribution with a small mean while the other is selected from another uniform distribution with a large mean. No correlation means that both weights are independently selected from uniform distributions. The primary cost of a link (i, j) is taken as $c(i, j) \sim uniform[1, 200]$. In addition to uniform distribution, we experiment with other distributions, including normal and exponential distributions.

To test the algorithms in the critical cases, the source and destination of a request are randomly generated such that the minimum hop-count between them is at least three. Note that one or two hop paths are easy to deal with since all such paths can be enumerated using a simple algorithm with at most the complexity of $\mathcal{O}(n^2)$. The constraints are also randomly generated, but their ranges are determined based on the best paths w.r.t. w_1 and w_2, as follows. Let p_1 and p_2 be two shortest paths from s to t w.r.t. w_1 and w_2, respectively. We take $c_1 \sim uniform[0.8w_1(p_2), 1.2w_1(p_2)]$ and $c_2 \sim uniform[0.8w_2(p_1), 1.2w_2(p_1)]$. The shaded box in Figure 8 represents the region from which the constraints

Link weight	Positive correlation	No correlation	Negative correlation
$w_1(i,j)$	$\sim uniform[1,50]$	$\sim uniform[1,100]$	$\sim uniform[1,50]$
$w_2(i,j)$	$\sim uniform[1,100]$	$\sim uniform[1,200]$	$\sim uniform[100,200]$
	OR		OR
$w_1(i,j)$	$\sim uniform[50,100]$		$\sim uniform[50,100]$
$w_2(i,j)$	$\sim uniform[100,200]$		$\sim uniform[1,100]$

TABLE I
RANGES OF LINK WEIGHTS AND THE CORRELATION BETWEEN THEM.

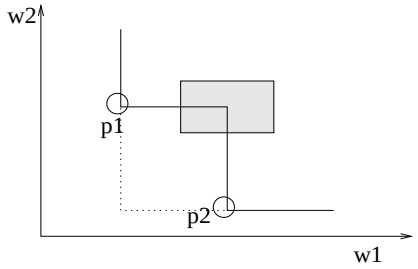

Fig. 8. The selection of constraints.

are selected. We extended this region in several directions (up, down, left, right). Although the absolute values of performance measures can change, the relative differences in the performance of compared algorithms do not change.

We contrast the performance of various path selection algorithms using two measures: (i) the *success ratio* (SR), which refers to the fraction of connection requests for which feasible paths are found by given algorithm, and (ii) the *average value of the primary cost function per routed connection* (AvgCost), where a *routed connection request* is one for which the given path selection algorithm returns a feasible path. AvgCost shows how costly a feasible path is, on average. The results reported in the subsequent sections are averaged over several runs and the 95% confidence intervals are computed. In each run, ten random graphs are generated. For each instance of a random graph, ten independent realizations of link weights are generated using different random seeds. Finally, for each instance of a random graph with given link weights (there are a total of 100 of such instances per experiment), about 2000, 2500, and 3000 connection requests are generated for graphs with 50, 100, and 200 nodes, respectively.

B. Performance of H_MCOP for Different Values of λ

From Corollary 1, it is expected that the performance of H_MCOP improves with λ. However, since H_MCOP is only an approximation of a nonexistent algorithm $\mathcal{X}$ that minimizes g_λ, our first goal is to verify that increasing λ will almost always improve the SR of H_MCOP, which gives the justification for using g^* (equivalently, h) in the design of H_MCOP. Figure 9 shows the average SR of H_MCOP versus λ for random graphs with 50, 100, and 200 nodes and with positively correlated, uncorrelated, and negatively correlated link weights. In particular, the difference in performance between the case of $\lambda = 1$ and $\lambda = \infty$ is quite significant. Due to the heuristic nature of the algorithm, one can expect few anomalies in the general trend. However, these deviations are observed to be negligible in magnitude and frequency.

C. Performance Comparison of Path Selection Algorithms

We now contrast the performance of H_MCOP (based on the cost function h) with other path selection algorithms. Figure 10 shows the SRs of various algorithms. When the link weights are positively correlated, the path weights also become positively correlated, and thus linear approximation algorithms have a good chance of finding feasible paths by minimizing the linear combination of link weights. However, if the link weights are negatively correlated, there will be more paths in the network for which $w_1(p) \gg w_2(p)$, or vice versa. This degrades the performance of Jaffe's linear approximation algorithms, which work best when the two link weights are positively correlated. In fact, by changing the slope of the search line (i.e., α and β) as done in Approx_2CP, the performance can be improved at the expense of more iterations of Dijkstra's algorithm. Although the worst-case complexity of Approx_2CP is $\log B$ times that of Dijkstra's algorithm, the actual number of calls to Dijkstra's algorithm varies in the range $[1, \log B]$, where B is the upper bound on the longest path w.r.t. one of the link weights. It should be noted that Approx_2CP runs at its worst-case complexity only if it is deemed to fail, i.e., if the algorithm succeeds in finding a path, it will do so with much fewer Dijkstra's iterations than $\log B$. A more detailed comparison of Approx_2CP with other algorithms is presented in [38]. To avoid the shortcomings of linear approximation algorithms, R_MCP and H_MCOP search for a feasible path using nonlinear cost functions and provide better SR than linear approximation algorithms. Note that H_MCOP requires at most two iterations of Dijkstra's algorithm and gives almost the same SR that of R_MCP and ER_MCP_D, both of which require $K + 2$ iterations of Dijkstra's algorithm. A more detailed study of R_MCP and its enhanced version is presented in [42]. To compare H_MCOP with Chen's heuristic, we need to properly set the value of x of the latter algorithm. As x goes to infinity, the SR of Chen's heuristic approaches that of the optimal exponential-time algorithm. But given its $\mathcal{O}(x^2n^2)$ complexity, a large x clearly makes the algorithm impractical.

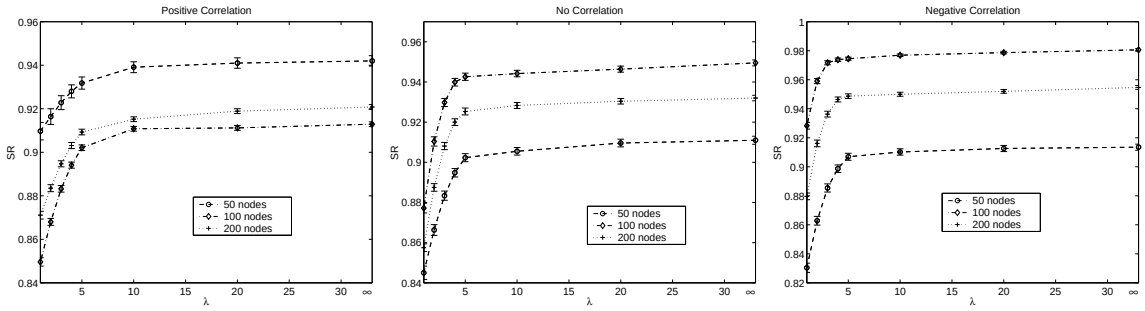

Fig. 9. Success rate of H_MCOP versus λ (95% confidence intervals are also depicted).

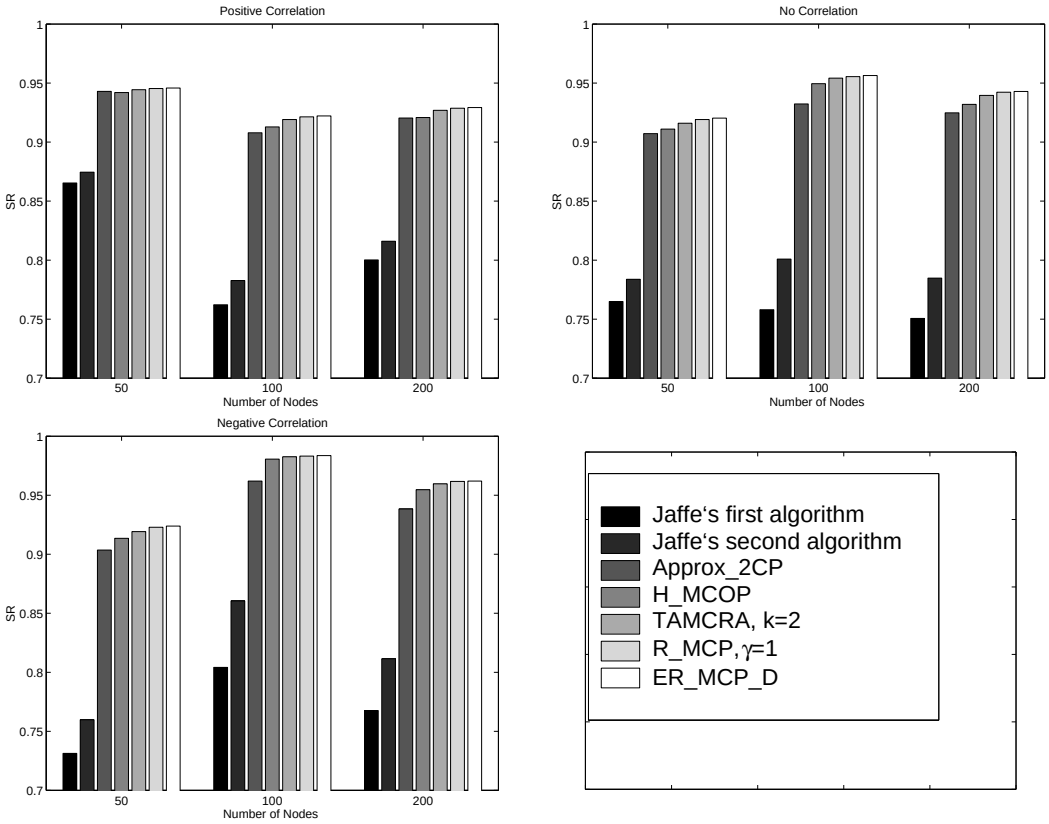

Fig. 10. SR of various algorithms used on random graphs with 50, 100, and 200 nodes.

To get as close as possible to achieving the same computational complexity of H_MCOP, x must be set to two. However, since in our simulations we consider paths with a minimum of three hops and Chen's algorithm finds paths with at most x hops, we set x to three. Even with $x = 3$, the SR of this algorithm still lags significantly behind others (its SR, which is not shown in the figure, is around 0.2). Note that with $x = 3$, Chen's algorithm requires nine iterations of Dijkstra, i.e., at least four times the running time of H_MCOP.

To compare H_MCOP with TAMCRA, we need to properly set the value of k in the latter algorithm. As k goes to infinity, the performance of TAMCRA approaches that of the exact exponential-time algorithm. But given its $\mathcal{O}(km\log(kn) + k^3m)$ complexity, a large k clearly makes the algorithm impractical. To get as close as possible to achieving the same computational complexity of H_MCOP, k must be set to one or two. Note that H_MCOP uses two iterations of the Dijkstra's algorithm to minimize the same nonlinear cost function that is used in TAMCRA, but TAMCRA uses a different heuristic based on k-shortest path algorithm. Figure 10 shows that H_MCOP has almost the same SR that of TAMCRA with $k = 2$ while the worst-case complexity of TAMCRA with $k = 2$ is slightly higher than that of H_MCOP. However, since both TAMCRA and H_MCOP can be used with k-shortest path algorithm and give better performance than other algorithms when k is increased, we single them out and contrast their performances in more detail in the next section. Although both algorithm gives the similar SR performance under comparable computational complexities, H_MCOP significantly reduces the cost of a selected feasible path over TAMCRA.

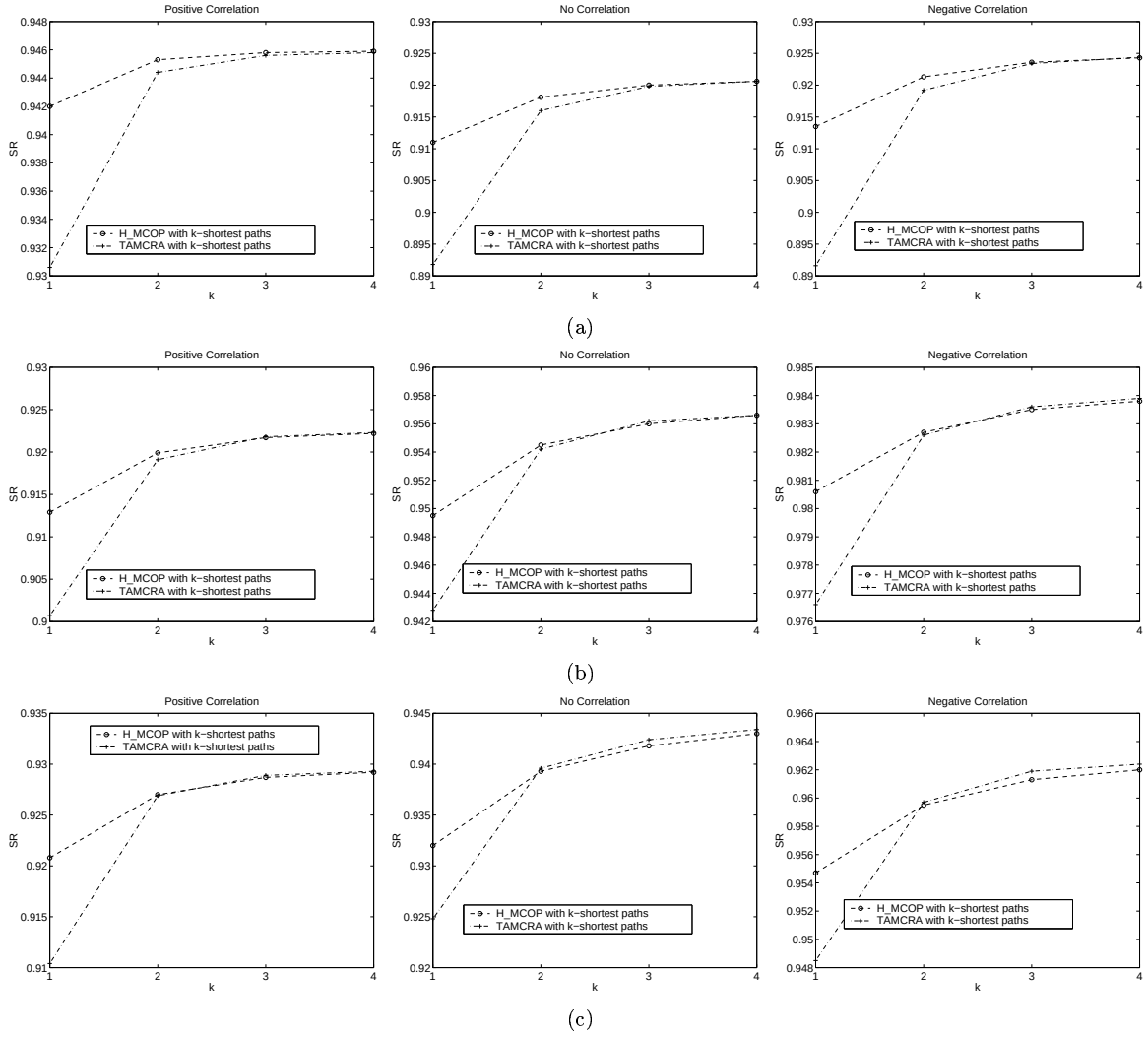

Fig. 11. SRs of H_MCOP and TAMCRA with k-shortest paths and with uniformly distributed link parameters: (a) 50 nodes; (b) 100 nodes; (c) 200 nodes.

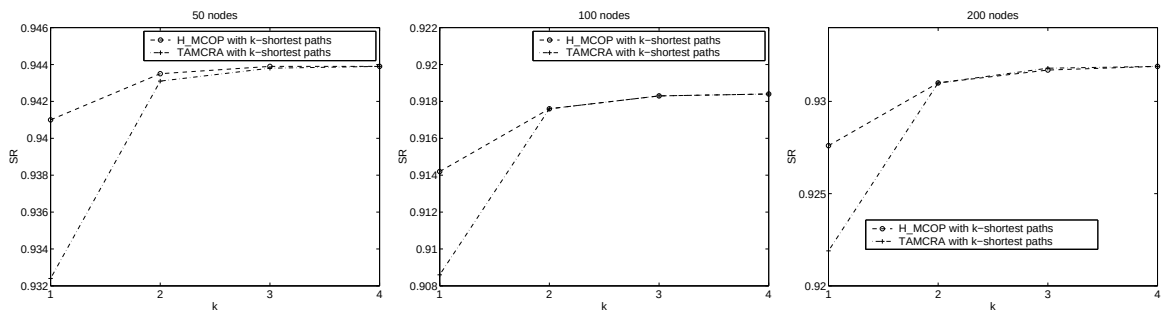

Fig. 12. SRs of H_MCOP and TAMCRA under normally distributed link parameters.

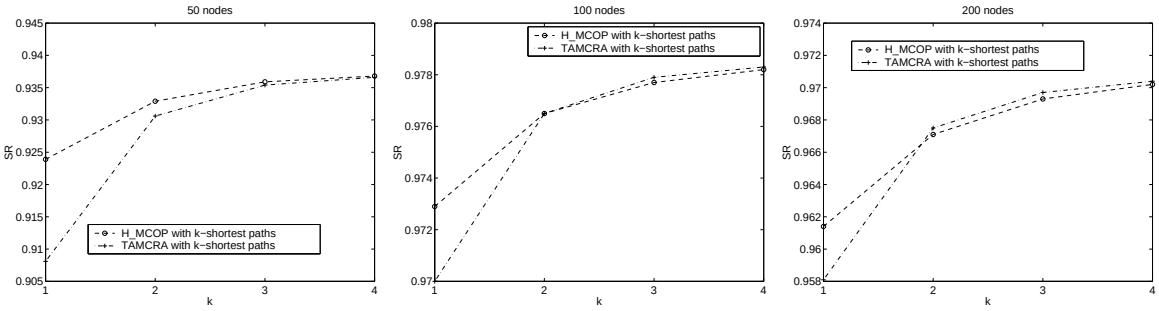

Fig. 13. SRs of H_MCOP and TAMCRA under exponentially distributed link parameters.

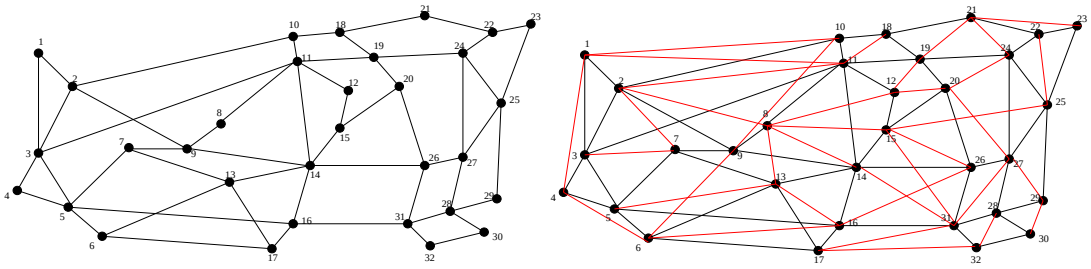

Fig. 14. Modified ANSNET topologies with 110 and 182 directed links.

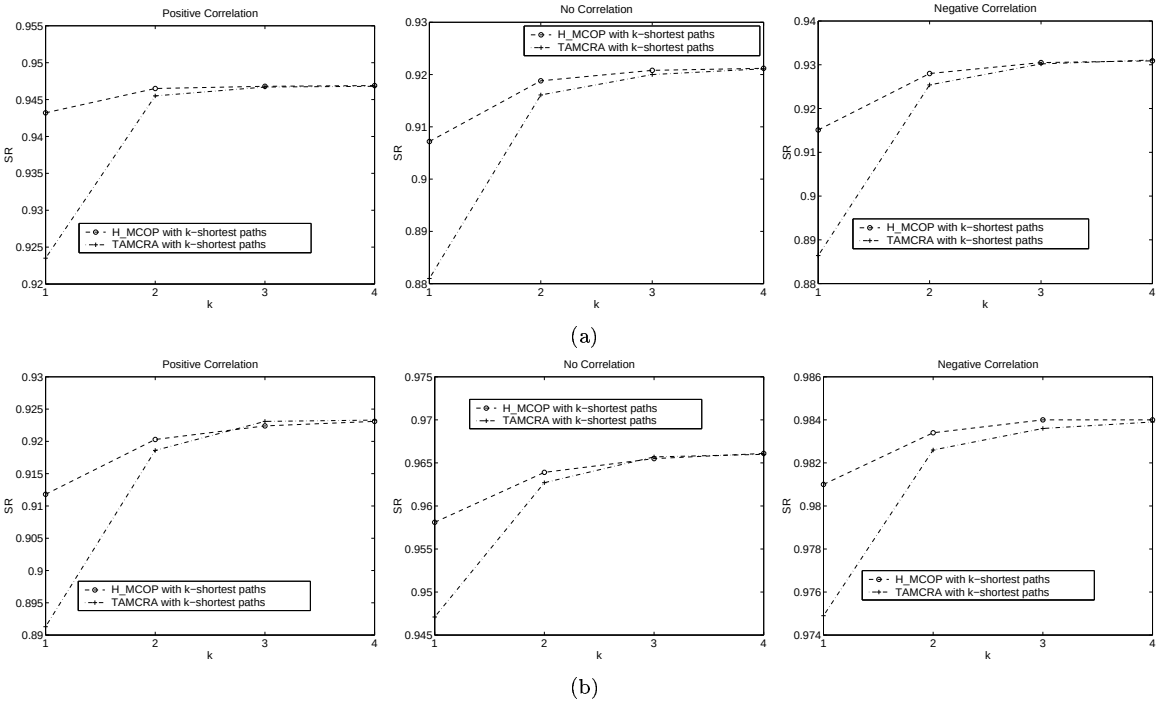

Fig. 15. SRs of H_MCOP and TAMCRA when ANSNET topologies are used with: (a) 110 links (b) 182 links.

D. Detailed Performance Comparison of H_MCOP and TAMCRA

In this section, we compare H_MCOP and TAMCRA in more detail, using both algorithms with k-shortest paths along with the eliminating dominated paths. We study several factors that impact the performance, including the number of nodes, the number of edges, the distribution of the link weights, the correlation between link weights, and the selection of constraints. We first consider the random topologies used before. In addition to the uniform distribution, we also consider normal and exponential distributions for link parameters. Figure 11 shows the SRs of both algorithms versus k when the link parameters are generated from the uniform distributions described in Section IV-A. Figure 12 shows the SRs of both algorithms when the link parameters are generated from normal distributions with the following means and standard deviations: $w_1(i,j) \sim normal(250, 80)$, $w_2(i,j) \sim normal(150, 50)$,

and $c(i,j) \sim normal(100,30)$. Figure 13 shows the SRs of both algorithms when the link parameters are generated from exponential distributions with the following means: $w_1(i,j) \sim exp(200)$, $w_2(i,j) \sim exp(150)$, and $c(i,j) \sim exp(200)$. In addition to random topologies, we consider two realistic topologies, shown in Figure 14. Both topologies have been modified from ANSNET [63] by inserting additional links. Figure 15 shows the SRs of both algorithms for these topologies with uniformly distributed link weights (similar trends have also been observed for other link-parameter distributions).

As shown in Figures 11 through 15, in general, H_MCOP gives slightly better SR than TAMCRA at the same values of k. The SR of both algorithm increases with k and finally converges to the SR of the exact, exponential-time algorithm (which, of course, cannot be obtained in practice as it requires enumerating all paths in the network). The same trend in SR performance is observed for various network sizes, distributions of link weights, and degrees of correlations between link weights. In Figures 11 through 15, we observe that the SR of H_MCOP (and of TAMCRA) for $k > 2$ is always high (e.g., larger than 0.92) and quite insensitive to the aforementioned factors. This is mainly attributed to the choices of the constraint values in the given graph (see Figure 8 and the explanations in Section IV-A). If the region from which the constraint values are selected is moved closer to or away from the origin of the 2-dimensional parameter space, the SR of the exact algorithm, and thus the SR of any heuristic algorithm, gets smaller or larger, respectively. The reason why we select the constraints in this fashion is that we needed to compare the contending algorithms in the (nontrivial) critical scenarios while still maintaining reasonable simulation times. Other, independently selected parameters (e.g., the size of the network, the distribution of link weights, and the correlation between link weights) cause some fluctuations in the SR, since they affect the distribution of paths in the 2-dimensional search space and also the size of the region from which the constraint values are sampled.

While the above comparisons demonstrate that H_MCOP and TAMCRA achieve the similar SR performance under comparable computational complexities (note that H_MCOP requires one additional iteration of Dijkstra's algorithm in the backward direction), the real advantage of H_MCOP is in terms of minimizing the cost of the returned feasible path (i.e., selecting a *resource-efficient* feasible path). This is demonstrated in Table II, which shows the percentage reduction in the AvgCost when H_MCOP is used instead of TAMCRA. Since TAMCRA does not perform path-cost optimization, it is unlikely that a path selected by TAMCRA will be cost effective. However, H_MCOP optimizes the cost parameter and achieves a significant cost reduction over TAMCRA, particularly when there are several feasible paths to consider. For example, adding more links to a topology increases the number of feasible paths. For example, reduction in AvgCost for the ANSNET topology with 182 links is larger than that for the ANSNET topology with 110 links. Another factor that affects the number of feasible paths is the correlation between link weights. For example, when link weights are positively correlated, the path weights are also positively correlated, and the best paths p_1 and p_2 get closer to each other. Consequently, the constraints are selected from a narrow region that does not contain many feasible paths. As a result, under positive correlation the cost reduction is less than the ones obtained under uncorrelated and negatively correlated link parameters. As path weights become less correlated, more feasible paths are added to the region from which the constraint values are selected, making the cost reduction achieved by H_MCOP quite significant. Again several factors including the size of the network, the correlation and distribution of link weights, and the selection of constraints affect the AvgCost of any algorithm since the optimal AvgCost differs depending on the variations in the above factors. However, simulation results indicate that H_MCOP always achieves a cost reduction ranging from 3.7% to 58.6% over TAMCRA. The results also show that using the k-shortest path algorithm in conjunction with H_MCOP does not significantly change the cost reduction. The reason is that most of the feasible paths are already found when $k = 1$, and thus increasing k makes the algorithm consider a few more feasible paths. But this is not sufficient to significantly change the AvgCost reduction.

V. Conclusions

Optimal path selection subject to multiple constraints is an NP-complete problem, which can only be addressed through heuristics and approximation algorithms. Previously proposed algorithms suffer from excessive computational complexities and/or low performance. Moreover, most of them are only applicable to special cases of the problem. In this paper, we investigated the general multi-constrained optimal path (MCOP) problem with the goal of developing highly efficient heuristics in terms of computational time, performance, and resource utilization. First, we investigated the theoretical properties of a nonlinear cost function, g_λ, that can be used as the basis for efficient heuristic solutions to the MCOP problem. We showed that as the nonlinearity parameter λ increases to infinity, the minimization of g_λ provides a better approximation to the MCP problem (the MCOP problem without path optimization). We demonstrated that a generalized linear approximation algorithm can be easily developed for $\lambda = 1$. For $\lambda > 1$, the nonlinearity of g_λ does not allow for an exactly polynomial path selection algorithm. Although finding a solution to the new minimization problem is left open, we provided an efficient heuristic algorithm (H_MCOP) that tries to approximate the minimization of $g_\lambda^* \stackrel{\text{def}}{=} \lim_{\lambda\to\infty} g_\lambda$. To optimize the use of resources while searching for a feasible path, H_MCOP also attempts to minimize a primary cost function. We proved that H_MCOP guarantees at least the same performance provided by a linear approximation algorithm, and most often provides significant improvements upon it. H_MCOP has the same order of complexity as Dijkstra's shortest path algorithm. For further

topology	distribution of link weights		k (the number of shortest paths) 1	2	3	4
ANSNET with 110 links	uniform	'+'	7.4%	8.3%	8.4%	8.4%
	uniform	'0'	13.7%	14.4%	14.6%	14.8%
	uniform	'−'	16.2%	16.9%	17.3%	17.4%
ANSNET with 187 links	uniform	'+'	17.9%	18.7%	19.1%	19.3%
	uniform	'0'	33.6%	34.4%	34.7%	35.0%
	uniform	'−'	36.5%	37.2%	37.7%	37.9%
Random with 50 nodes	uniform	'+'	6.8%	7.4%	7.5%	7.5%
	uniform	'0'	12.1%	13.1%	13.4%	13.5%
	uniform	'−'	14.9%	15.7%	16.1%	16.3%
	normal	'0'	3.7%	3.9%	4.0%	4.0%
	exponential	'0'	29.0%	29.8%	30.1%	30.3%
Random with 100 nodes	uniform	'+'	18.4%	19.2%	19.5%	19.6%
	uniform	'0'	34.6%	35.2%	35.5%	35.7%
	uniform	'−'	38.0%	38.4%	38.6%	38.7%
	normal	'0'	8.5%	8.8%	8.9%	8.9%
	exponential	'0'	57.9%	58.3%	58.5%	58.6%
Random with 200 nodes	uniform	'+'	14.0%	14.7%	14.9%	15.0%
	uniform	'0'	25.7%	26.3%	26.7%	27.1%
	uniform	'−'	27.9%	28.4%	28.7%	28.8%
	normal	'0'	6.0%	6.3%	6.4%	6.4%
	exponential	'0'	48.9%	49.2%	49.4%	49.6%

TABLE II

PERCENTAGE REDUCTION IN AVGCOST WHEN H_MCOP IS USED INSTEAD OF TAMCRA. THE LABELS '+', '−', AND '0' INDICATE POSITIVE, NEGATIVE, AND NO CORRELATION BETWEEN LINK WEIGHTS, RESPECTIVELY.

performance improvement, H_MCOP can also be used in conjunction with the k-shortest path algorithm.

Using extensive simulations, we first verified that the performance of H_MCOP generally improves with λ, with few anomalies that are negligible in magnitude and frequency. We then contrasted H_MCOP with other contending algorithms. Our results show that at a fixed computational complexity, H_MCOP outperforms existing algorithms in terms of the SR, followed by TAMCRA (which also uses g^*_λ as a basis for path selection). Consequently, we compared H_MCOP with TAMCRA in more detail using the k-shortest paths in both and link weights generated from different distributions including uniform, normal, and exponential. The results indicate that a much larger value of k (i.e., more computational time) is needed in TAMCRA to produce the same SR obtained through H_MCOP regardless of network size, correlation and distribution of link weights. Furthermore, H_MCOP reduces the AvgCost of the returned feasible paths over TAMCRA, thus providing more efficient use of network resources. The AvgCost reduction is significant when the network contains a large number of feasible paths. We also investigated the impact of correlated and uncorrelated link weights on the path selection algorithms. We observed that when the path weights are negatively correlated, i.e., $w_1(p) \gg w_2(p)$, or vice versa, linear approximation algorithms often return such paths that satisfy one constraint but violets the other. When link weights are positively correlated, linear approximation algorithms are more likely to succeed in finding feasible paths. The simulation results verified that when the link weights are positively correlated, the path weights also become positively correlated and thus the linear approximation algorithms often succeeds in finding feasible paths. However, when link weights are negatively correlated, the path weights tend to also be negatively correlated, degrading the performance of linear approximation algorithms. In all cases, H_MCOP was shown to provide better performance than linear approximation algorithms. When negative or no correlations are present between link weights, H_MCOP provides significant performance improvement upon linear approximation algorithms than when positive correlation exists.

H_MCOP performs well when the true state of the network is given. However, the true state may not be available to every node at all times due to network dynamics, aggregation of state information, and latencies in state dissemination. As a future work, we will investigate how H_MCOP performs in the presence of inaccurate state information and what modifications need to be done.

REFERENCES

[1] S. V. Raghavan and S. K. Tripathi, *Networked Multimedia Systems: Concepts, Architecture, and Design*, Prentice Hall, Inc., 1998.

[2] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of service based routing: A performance perspective," in *Proceedings of the ACM SIGCOMM '98 Conference*, Van-

couver, British Columbia, Canada, August-September 1998, pp. 17–28.

[3] S. Chen and K. Nahrstedt, "An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions," *IEEE Network*, vol. 12, no. 6, pp. 64–79, Nov-Dec 1998.

[4] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A framework for QoS-based routing in the Internet," Tech. Rep. RFC 2386, IETF, August 1998.

[5] R. Guerin and A. Orda, "Networks with advance reservations: the routing perspective," in *Proceedings of the INFOCOM 2000 Conference*. IEEE, 2000, vol. 1, pp. 118–127.

[6] R. Vogel, R.G. Herrtwich, W. Kalfa, H. Wittig, and L.C. Wolf, "QoS-based routing of multimedia streams in computer networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1235–1244, September 1996.

[7] X. Xiao and L. M. Ni, "Internet QoS: A big picture," *IEEE Network*, vol. 13, no. 2, pp. 8–18, March-April 1999.

[8] The ATM Forum, "Private network-to-network interface specification version 1.0 (PNNI 1.0)," March 1996, af-pnni-0055.000.

[9] T. M. Chen and T. H. Oh, "Reliable services in MPLS," *IEEE Communication Magazine*, vol. 37, no. 12, pp. 58–62, Dec. 1999.

[10] J. Moy, "OSPF version 2," Standards Track RFC 2328, IETF, April 1998.

[11] G. Apostolopoulos, D. Williams, S. Kamat, A. Guerin, R. Orda, and T. Przygienda, "QoS routing mechanisms and OSPF extensions," Tech. Rep. RFC 2676, IETF, August 1999.

[12] R. Guerin and A. Orda, "QoS routing in networks with inaccurate information: Theory and algorithms," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 350 –364, June 1999.

[13] A. Alles, "ATM internetworking," White Paper, Cisco Systems, Inc., May 1995.

[14] Z. Wang, "On the complexity of quality of service routing," *Information Processing Letters*, vol. 69, no. 3, pp. 111–114, 1999.

[15] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, September 1996.

[16] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Inc., 1993.

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[18] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, vol. 14, pp. 95–116, 1984.

[19] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations Research*, vol. 17, no. 1, pp. 36–42, 1992.

[20] R. Widyono, "The design and evaluation of routing algorithms for real-time channels," Tech. Rep. TR-94-024, University of California at Berkeley & International Computer Science Institute, June 1994.

[21] C. A. Phillips, "The network inhibition problem," in *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC)*, May 1993, pp. 776–785.

[22] A. Goel, K.G. Ramakrishnan, D. Kataria, and D. Logothetis, "Efficient computation of delay-sensitive routes from one source to all destinations," in *Proceedings of the INFOCOM 2001 Conference*. IEEE, April 2001, vol. 2, pp. 854–858.

[23] F. Ergun, R. Sinha, and L. Zhang, "QoS routing with performance-dependent costs," in *Proceedings of the INFOCOM 2000 Conference*. IEEE, 2000, vol. 1, pp. 137–146.

[24] A. Orda, "Routing with end-to-end QoS guarantees in broadband networks," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 365–374, 1999.

[25] D. H. Lorenz, A. Orda, D. Raz, and Y. Shavitt, "Efficient QoS partition and routing of unicast and multicast," in *IWQoS 2000*, June 2000, pp. 75–83.

[26] D. Eppstein, "Finding the k shortest paths," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. IEEE, Nov. 1994, pp. 154 – 165.

[27] L. Guo and I. Matta, "Search space reduction in QoS routing," in *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*. IEEE, May 1999, pp. 142 – 149.

[28] G. Y. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, pp. 293–310, 1980.

[29] Y. P. Aneja, V. Aggarwal, and K. P. K. Nair, "Shortest chain subject to side constraints," *Networks*, vol. 13, pp. 295–302, 1983.

[30] D. S. Reeves and H. F. Salama, "A distributed algorithm for delay-constrained unicast routing," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 239–250, April 2000.

[31] Q. Sun and H. Langendorfer, "A new distributed routing algorithm for supporting delay-sensitive applications," *Computer Communications*, vol. 21, pp. 572–578, 1998.

[32] K. Ishida, K. Amano, and N. Kannari, "A delay-constrained least-cost path routing protocol and the synthesis method," in *Proceedings of the Fifth International Conference on Real-Time Computing Systems and Applications*. IEEE, Oct. 1998, pp. 58 – 65.

[33] I. Cidon, R. Rom, and Y. Shavitt, "Multi-path routing combined with resource reservation," in *Proceedings of the INFOCOM '97 Conference*. IEEE, 1997, pp. 92 – 100.

[34] R. Sriram, G. Manimaran, and C. S. R. Murthy, "Preferred link based delay-constrained least-cost routing in wide area networks," *Computer Communications*, vol. 21, pp. 1655–1669, 1998.

[35] Y.P. Aneja and K.P.K. Nair, "The constrained shortest path problem," *Naval Research Logistics Quarterly*, vol. 25, pp. 549–555, 1978.

[36] D. Blokh and G. Gutin, "An approximate algorithm for combinatorial optimization problems with two parameters," *Australasian Journal of Combinatorics*, vol. 14, pp. 157–164, 1996.

[37] A. Juttner, B. Szviatovszki, I. Mecs, and Z. Rajko, "Lagrange relaxation based method for the QoS routing problem," in *Proceedings of the INFOCOM 2001 Conference*. IEEE, April 2001, vol. 2, pp. 859–868.

[38] T. Korkmaz, M. Krunz, and S. Tragoudas, "An efficient algorithm for finding a path subject to two additive constraints," *Computer Communications Journal*, vol. 25, no. 3, pp. 225–238, 2002.

[39] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," in *Proceedings of the ICC '98 Conference*. IEEE, 1998, pp. 874 –879.

[40] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," Tech. Report UIUCDCS-R-97-2026, Dept. of Computer Science, University of Illinois at Urbana-Champaign, August 1997.

[41] X. Yuan, "On the extended Bellman-Ford algorithm to solve two-constrained quality of service routing problems," in *Proceedings of the International Conference on Cmputer Communications and Networks*. IEEE, 1999, pp. 304 –310.

[42] T. Korkmaz and M. Krunz, "A randomized algorithm for finding a path subject to multiple QoS constraints," *Computer Networks Journal*, vol. 36, no. 2-3, pp. 251–268, 2001.

[43] H. De Neve and P. Van Mieghem, "A multiple quality of service routing algorithm for PNNI," in *Proceedings of the ATM Workshop*. IEEE, May 1998, pp. 324 – 328.

[44] E. I. Chong, S. R. Sanjeev Rao Maddila, and S. T. Morley, "On finding single-source single-destination k shortest paths," in *the Seventh International Conference on Computing and Information (ICCI '95)*, July 5-8, 1995, pp. 40–47.

[45] Z. Wang and J. Crowcroft, "Bandwidth-delay based routing algorithms," in *Proceedings of the GLOBECOM '95 Conference*. IEEE, Nov. 1995, vol. 3, pp. 2129–2133.

[46] Q. Ma and P. Steenkiste, "Routing traffic with quality-of-service guarantees in integrated services networks," in *Proceedings of NOSSDAV '98*, July 1998.

[47] S. Verma, R. K. Pankaj, and A. Leon-Garcia, "QoS based multicast routing algorithms for real time applications," *Performance Evaluation*, vol. 34, no. 4, pp. 273–294, 1998.

[48] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-service routing using maximally disjoint paths," in *the Seventh International Workshop on Quality of Service (IWQoS '99)*, London, England, May/June 1999, IEEE, pp. 119 – 128.

[49] M. Kodialam and T. V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proceedings of the INFOCOM 2000 Conference*. IEEE, 2000, vol. 2, pp. 902–911.

[50] A. Orda and A. Sprintson, "QoS routing: the precomputation perspective," in *Proceedings of the INFOCOM 2000 Conference*. IEEE, 2000, vol. 1, pp. 128–136.

[51] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proceedings of the INFOCOM 2000 Conference*. IEEE, 2000, vol. 2, pp. 519–528.

[52] W. C. Lee, M. G. Hluchyi, and P. A. Humblet, "Routing subject to quality of service constraints in integrated communication networks," *IEEE Network*, pp. 46–55, July/August 1995.

[53] A. Iwata, R. Izmailov, B. Lee, D-S. Sengupta, and H. Ramamurthy, G. Suziki, "ATM routing algorithms with multiple QOS requirements for multimedia internetworking," *IEICE Trans. Commun.*, vol. E79-B, no. 8, pp. 999–1006, August 1996.

[54] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP '97)*, 1997, pp. 191 –202.

[55] C. Pornavalai, G. Chakraborty, and N. Shiratori, "QoS based routing algorithm in integrated services packet networks," in *Proceedings of ICNP '97*. IEEE, 1997, pp. 167–174.

[56] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proceedings of the INFOCOM '94 Conference*, 1994, vol. 2, pp. 636–646.

[57] I. Stoica and H. Zhang, "Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks," in *Proceedings of the ACM SIGCOMM '99 conference on Applications, technologies, architectures, and protocols for computer communication*, 1999, pp. 81–94.

[58] J. C. R. Bennett and H. Zhang, "WF^2Q: worst-case fair weighted fair queueing," in *Proceedings of the INFOCOM '96 Conference*, 1996, vol. 1, pp. 120–128.

[59] D. Clark and J. Joseph Pasquale, "Strategic directions in networks and telecommunications," *ACM Computing Surveys*, vol. 28, no. 4, pp. 579–690, 1996.

[60] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT press and McGraw-Hill book company, sixteenth edition, 1996.

[61] K. I. Calvert, M. B. Doar, and E. W. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160 – 163, June 1997.

[62] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 69, pp. 1617–1622, Dec. 1988.

[63] D. E. Comer, *Internetworking with TCP/IP*, vol. I, Prentice Hall, Inc., third edition, 1995.